Hierarchical Task Network and Operator-Based Planning: Two Complementary Approaches to Real-World Planning

Tara A. Estlin, Steve A. Chien, and Xuemei Wang

Jet Propulsion Laboratory

California Institute of Technology

4800 Oak Grove Drive, M/S 126-347

Pasadena, CA 91109-8099

Contact Author: Tara Estlin

Email: tara.estlin@jpl.nasa.gov

Abstract

Work on generative planning systems has focused on two diverse approaches to plan construction. Hierarchical Task Network (HTN) planners build plans by successively refining high-level goals into lower-level activities. Operator-based planners employ means-end analysis to directly formulate plans consisting of low-level activities. While many have argued the universal dominance of a single approach, we present an alternative view: that in different situations either may be most appropriate. To support this view, we describe a number of advantages and disadvantages of these approaches in light of our experiences in developing two real-world, fielded planning systems.

1 Introduction

AI planning researchers have developed numerous approaches to the task of correct and efficient planning. Two main approaches to generative planning are *operator-based* planners and *hierarchical task network* (HTN) planners. While considerable work has been done in analyzing and formalizing each of these approaches [Chapman 1987, Erol et al. 1994], and some work has been done in comparing these approaches from a theoretical standpoint

[Kambhampati 1995, Minton et al. 1991], comparatively little effort has been devoted to comparing the two approaches in a more practical setting.

While both HTN and operator-based planners typically construct plans by searching in a plan-space, they differ considerably in how they express plan refinement operators. HTN planners generally specify plan modifications in terms of flexible task reduction rules. Operator-based planners perform all reasoning at the lowest level of abstraction and provide a strict semantics for defining operator definitions. By virtue of their representation, HTN planners more naturally represent hierarchy and modularity. In contrast, operator-based plan refinements are more general in nature since they can cover many more planning situations.

In this paper, we explain how a hybrid approach, which combines these two planning techniques, is an effective method for planning in real-world applications. In particular, we investigate the critical issue of planning representation. If domain knowledge can be naturally represented in a planning system, then: (1) it will be easier to encode an initial knowledge base; (2) fewer encoding errors will occur, leading to a higher performance system; and (3) maintenance of the knowledge base will be considerably easier. Thus, an important measure for evaluating HTN and operator-based planning is how naturally each paradigm can represent key aspects of planning knowledge.

To evaluate representation abilities, we focus on four criteria: generality, hierarchy, flexibility, and efficiency. Generality describes the range of problem-solving situations that can be covered by a small amount of represented knowledge. Hierarchies allow common constraints, procedures, and patterns to be described in a single place yet used many times and are essential for efficient knowledge maintenance. Flexibility describes how easily a wide range of constraints can be accurately represented. Efficiency relates to how the representation influences the size of the planner's search space.

This paper describes a number of important representational issues concerning these evaluation criteria which we have encountered in building planning systems for two NASA application areas: Image Processing for Science Data Analysis (the MVP system) [Chien 1994, Chien et al. 1995] and Deep Space Network Antenna Operations (DPLAN) [Chien et al. 1997, Chien et al. 1995]. The Multimission VICAR Planner (MVP) uses a combination of planning techniques to automatically generate image processing programs from user-specified processing goals. MVP allows a user to specify a list of image-processing requirements in terms of necessary image corrections. Given this information, MVP derives the required processing steps and constructs a set of complex executable procedures, which will achieve the input processing goals. Our second application involves scheduling Deep Space Network (DSN) Antennas. In this domain, the DPLAN planner is used to generate a list of antenna operation steps that will create a communications link with an orbiting spacecraft. DPLAN inputs a set of antenna tracking goals and other relevant information such as available equipment types and required equipment settings. DPLAN's output is a track-specific temporal dependency network (TDN) which is used by an execution agent to automatically conduct the requested antenna operations.

Both of the planners described above employ a similar combination of HTN and operator-based planning techniques. Constructing and experimenting with these systems has helped us to closely examine many of the representation and efficiency trade-offs generated when using an integrated planning framework.

2 Integrating Planning Techniques

While we presume that the reader has a working knowledge of basic operator-based planning and HTN planning techniques, we briefly review the most salient differences of the two approaches and discuss how they can be effectively integrated in one planning framework. We also discuss two other related systems which also use some combination of HTN and operator-based planning techniques.

2.1 An Overview of HTN and Operator-based Planning

An HTN planner [Erol et al. 1994] uses task reduction rules to decompose abstract goals into lower level activities. A set of constraints is maintained similar to those found in an operator-based planner (e.g., effects, causal protections) and many of the same methods can be used to resolve possible conflicts. HTN planners can encode many types of information into task reductions. By defining certain reduction refinements, the designer can direct the planner towards particular search paths in certain contexts. The user can also directly influence the planner by explicitly adding an ordering constraint or goal protection that would not strictly be derived from goal interaction analyses. In addition, search-control knowledge can usually be easily encoded by writing explicit action sequences to achieve goals, thereby avoiding considerable search. HTN planners are thus considered very flexible in representing domain information; however, this flexibility can often lead to numerous overly-specific reduction rules that can be difficult to understand and maintain.

In contrast, an operator-based planner reasons at a single level of abstraction - the lowest level (e.g. [Penberthy and Weld 1992, Carbonell et al. 1992]). Actions are strictly defined in terms of preconditions and effects; plans are produced through subgoaling and goal interaction analysis. In this framework, all plan constraints (protections, ordering, codesignation) are a direct consequence of goal achievements and action precondition and effect analysis. Thus, an operator-based planner generally has a strict semantics grounded in explicit state representation, i.e. defining what is and is not true in a particular state (or partial state). This rigid representation is both a strength and a weakness. It is advantageous since it more explicitly directs the knowledge engineer in encoding a domain. Yet, it can also make certain aspects of a problem difficult to represent. For example, known ordering constraints and operator sequences can be difficult to encode if they cannot easily be represented in terms of preconditions and effects. Such constraints can and are often forced by adding "dummy" preconditions, however this solution can often create a misleading representation.

2.2 An Integrated Planning Framework

In an integrated HTN/operator framework, a planner can use multiple planning methods and reason about different types of planning goals. For example, in DPLAN, we have defined two types of goals: *activity-goals* and *state-goals*. *Activity-goals* correspond to operational

Let R be a set of decomposition rules,

Q be a list of partial-plans,

P be the current plan, and

g be an unachieved goal in P,

If *g* is an Activity-Goal,

1. Decompose: For each decomposition rule r in R which can decompose *g*, apply r to produce a new plan P', If all constraints in P' are consistent, then add P' to Q.
2. Simple Establishment: For each other activity-goal *g'* in P that can be unified with *g*, simple establish *g* using *g'* and produce a new plan P'. If all constraints in P' are consistent, then add P' to Q.

If *g* is a State-Goal,

1. Step Addition: For each activity-goal, *g'* that asserts *g* as an effect, add *g'* to P to produce a new plan P'. If the constraints in P' are consistent, then add P' to Q.
2. Simple Establishment: For each activity-goal *g'* in U that has an effect e that can be unified with *g*, simple establish *g* using e and produce a new plan P'. If all constraints in P' are consistent, then add P' to Q.

Figure 1: Goal Refinement Strategies

or non-operational domain activities and are manipulated using HTN planning techniques. Operational activity-goals are considered primitive tasks that can be directly executed. Non-operational activity-goals must be further decomposed into operational ones through reduction rules. *State-goals* are defined as the preconditions and effects of activity-goals, and are achieved through standard operator-based planning methods. In our framework, a top-level goal specification consists of a set of abstract non-operational activity-goals and a set of unachieved state-goals. Figure 1 shows the procedures used for refining these two types of goals. As soon as a refinement strategy is applied to an activity-goal or state-goal, it is removed from a plan's list of unachieved goals. Planning is complete when all activity-goals are operational and all state-goals have been achieved. A planning solution consists of an ordered list of operational activity-goals.

2.3 Related Work

Much practical work in AI planning systems has been done in the context of decompositional planners. Two very related systems to MVP and DPLAN are SIPE [Wilkins 1988] and O-Plan [Tate et al. 1994][1]. Both of these systems support domain-independent planning of hierarchical and partial-order plans. The SIPE planning system was originally developed to produce hierarchical plans that could contain parallel actions. O-Plan builds upon the idea of task networks which are used to expand abstract plans into more detailed sets of plan steps and constraints.

While SIPE and O-Plan allow for integration of both HTN and operator-based planning, neither SIPE nor O-Plan make a direct distinction between HTN and operator-based planning techniques. Instead, plan formulation is primarily done using decomposition operators (or networks). Operator-based features such as preconditions and effects are added to these structures when necessary. In contrast, we argue for an approach in which HTN planning and operator-based techniques can be used in conjunction *or* as separate planning methods. Domain information pertaining to these two techniques is also kept separate; decompositional information is specified in decomposition rules, while items such as activity precondition and effects are kept in a separate schema list. This distinction is intended to allow a planner to apply a wider variety of planning techniques and to formulate domain information more rigorously in a flexible and usable representation.

3 Representing Hierarchical and Modularity Information

Many of the obstacles in applying planning techniques to real-world problems can be characterized as representation difficulties. Correctly representing complex actions and other detailed domain information can be a challenging problem in many domains. One advantage

[1]It is worth noting that these systems comprise 4 of the 5 applications described in [IEEE Expert 1996].

to employing HTN planning techniques is the ability to use abstract representation levels of domain objects and goals. Domain objects are usually items that can be manipulated through domain rules, such as blocks in the blocksworld. Goals can be either decomposition goals, user-specified goals, or activity preconditions that need to be achieved. Allowing abstract representations of these items enables us to represent domains in an object-oriented form, which is easier to write and reason about. This format also contributes to a more general domain knowledge base that can be efficiently updated and maintained.[2]

Unfortunately, this type of generalized format makes it difficult to represent specialized constraint information. Though we would like our domain information to be as modular as possible, there often exists a few pieces of detailed information that make this difficult. By incorporating operator-based planning techniques, we have been able to retain a modular domain format, while still correctly representing more specialized constraints.

3.1 Object and Goal Hierarchies

When using an HTN planner, different abstract levels of domain objects and goals can be represented by constructing an object or goal hierarchy. More detailed information such as object instances is at one end of a hierarchy, while very general information such as broad object types is at the other end.

In the DSN domain, different types of equipment are often required for separate antenna activities. For example, several types of antennas are represented in our domain, which have different attributes, such as size and type. Additionally, even two antennas of the same type are not identical. For instance, two 34 Meter Beam Wave Guide (BWG) antennas that are located at different stations may need to be treated differently. Our domain also includes several different types of receivers such as the Block-IV and Block-V receivers. These receivers are used to receive data transmissions from orbiting spacecraft. In Figure 2 we show partial equipment hierarchies for antennas and receivers.

[2]For a discussion of these issues in the context of representing reactive planning knowledge see [Firby 1996].

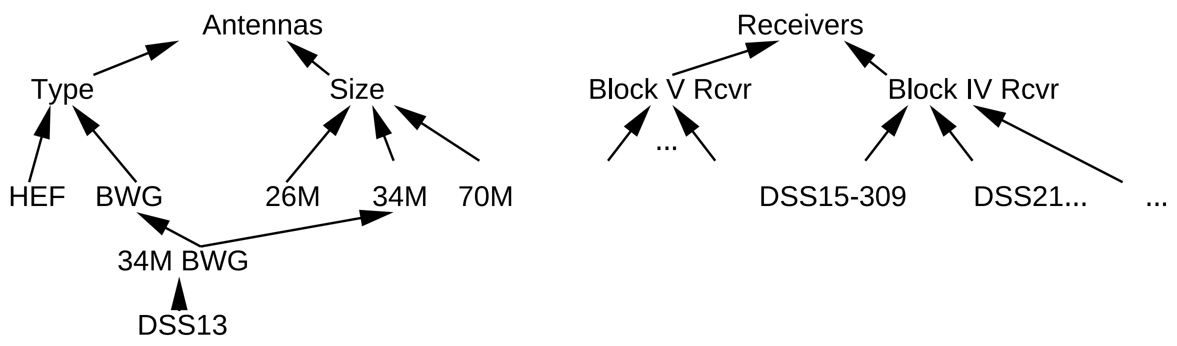

Figure 2: Antenna and Receiver Hierarchies

The main advantage to this type of representation is that decomposition rules can refer to either low- or high-level forms of a particular object or goal. In the Deep Space Network domain, a common antenna operation is performing a telemetry (or downlink) pass where information is transmitted from a spacecraft to an antenna. A telemetry pass usually requires one of several types of receivers depending on the particular antenna being used. The main steps of a telemetry pass may be very similar for different size antennas even though different receiver types are required. By using object and goal hierarchies we can write just one telemetry decomposition rule to represent the general steps taken during this operation. For instance, a decomposition rule for a telemetry track is shown in Figure 3. In this rule a general *perform-receiver-configuration* goal is asserted as a new goal.

Information pertaining to specific equipment is contained in smaller, more specialized rules. For instance, specific receiver configuration steps can be added separately by decomposing the *perform-receiver-configuration* goal into more specialized goals or low-level plan steps. The rules listed in Figure 4 show two possible ways to break down this goal. The first rule states that if the current goal is to configure the receiver, and the receiver assigned to this antenna track is a Block-IV receiver, then the configuration method for Block-IV receivers should be used. The second rule states a similar method for Block-V receivers. By using a general *perform-receiver-configuration* goal in the telemetry track decomposition rule, we avoid writing multiple versions of this general rule. Instead, specific receiver information is propagated to smaller, more specialized rules.

```
(decomprule default-telemetry-track
   lhs
     (initialgoals  ((track-goal spacecraft-track telemetry ?track-id)))
   rhs
     (newgoals   ((g1 (perform-antenna-controller-configuration ?track-id))
                  (g2 (perform-exciter-and-transmitter-configuration ?track-id))
                  (g3 (perform-microwave-controller-configuration ?track-id))
                  (g4 (perform-receiver-configuration ?track-id))
                  (g5 (perform-telemetry-configuration ?track-id))
                  (g6 (move-antenna-to-point ?track-id))
                  (g7 (perform-receiver-calibration ?track-id)))
      constraints ((before g1 g6)
                  (before g7 g3)
                  (before g4 g7))))
```

Figure 3: Telemetry Decomposition Rule

```
(decomprule default-configure-receiver1
   lhs
     (initialgoals  ((perform-receiver-configuration ?track-id))
      conditions  ((CCN-equipment-assignment ?track-id ?equip)
                   (isa ?equip BLOCK-IV-RECEIVER)))
   rhs
     (newgoals   ((configure-block-iv-receiver ?track-id ?equip))))


(decomprule default-configure-receiver2
   lhs
     (initialgoals  ((perform-receiver-configuration ?track-id))
      conditions  ((CCN-equipment-assignment ?track-id ?equip)
                   (isa ?equip BLOCK-V-RECEIVER)))
   rhs
     (newgoals   ((configure-block-v-receiver ?track-id ?equip))))
```

Figure 4: Two Decomposition Rules for Receiver Configuration

There are many benefits to this type of representation. By allowing object and goal hierarchies, we can construct domains in an object-oriented approach. This format allows domain information to be stored in a more compact and usable representation. If our representation did not allow these hierarchies, we would be forced to incorporate a large amount of specialized information into many of our decomposition rules. This would also cause many large and redundant rules to be defined. For instance, in the DSN domain there are four different receiver types: Block-III, Block-IV, Block-V, and MFR. In order to correctly represent specialized receiver information about each of these types, we would need to write four different telemetry rules. Different equipment combinations could also cause more rules to be added. For the 34 Meter antennas alone, there are three different antenna types. It's probable that some antenna/receiver combinations could act differently and thus require special steps to be added to any telemetry operation which used that particular pair of equipment. Therefore, we could possibly need 12 different telemetry rules, each for a different antenna/receiver combination. Additionally, there could be other specific goal interactions that could add even more steps to a rule in certain contexts. These interactions could even further increase the number of required telemetry rules. Rules for other types of tracking goals may also be affected similarly. By incorporating object and goal hierarchies we can remove this specialized equipment information from high-level rules, such as the telemetry rule shown in Figure 3, and allow them to just have one general definition. Specialized information is instead kept in smaller, more low-level rules and only accessed as necessary.

Domain information is also more easily understood and updated in this format since domain details are kept separate from more general knowledge. For example, to understand the general steps of a telemetry operation, a user only has to view the main telemetry track decomposition rule. If more low-level knowledge is desired, such as how to operate a particular piece of equipment, the user could then search for rules that directly pertain to that equipment type. Knowledge maintenance is also more efficient in this format. Most domain updates involve changes to only low-level steps. For instance, a new type of receiver could be added in the DSN domain, and therefore a new receiver configuration method might also be included. In our representation, this type of domain change need not affect more general domain information, such as the main telemetry rule. In other words, this update would not cause any rules that refer to the general *perform-receiver-configuration* to be modified; only a few more specialized rules would need to be created or updated.

Unlike operator-based planners, HTN planning algorithms provide direct support for this type of abstract representation. HTN planners are designed to operate at different abstract levels, where low-level details are kept separate from more general information. Operator-based planners perform all reasoning at the lowest level, and do not distinguish between different abstract levels of domain knowledge. Though some operator-based planners do allow object type hierarchies, they are not used for high-level decision making.[3] Thus, a major benefit of utilizing HTN planning techniques, is the ability to explicitly represent abstract domain information. We feel this type of representation flexibility contributes to the understandability of our domain and provides for more efficient maintenance operations.

3.2 Modularity vs. Specialized Constraints

Though it is advantageous to keep domain information in a general and modular format, this type of representation often makes it difficult to represent more specialized inter-modular constraints. These types of constraints refer to information inside of several different decomposition rules and are usually only applicable in certain situations. Adding these constraints forces the addition of more specialized rules and often causes a hierarchical representation of rules to be infeasible.

For example, in the DSN domain, a common activity is to perform receiver calibration. Similar to our receiver-configuration example in the previous section, we would like to define a general *calibrate-receiver* goal which is referred to by upper-level rules, and then use several corresponding lower-level calibration goals for the different receiver types, e.g. *calibrate-*

[3]Instead, their use is intended for making variable binding decisions, such as what object instances can be bound to a particular variable.

block-iv-receiver, *calibrate-block-v-receiver*, etc. In most antenna tracks, it is possible to refer to the more abstract *calibrate-receiver* goal in the higher-level rules, however, when using the Block-IV receiver, VLBI (Very Long Baseline Interferometry) telemetry tracks directly impose high-level ordering constraints on specific Block-IV calibration steps, instead of on the more general *calibrate-receiver* goal.

Two different VLBI tracks are displayed in Figure 5. The left track uses a Block-V receiver while the right one employs a Block-IV receiver. Low-level receiver calibration steps are shown in the shaded areas. In the Block-V receiver case, configuration is mapped onto a single operator; in the Block IV receiver case it corresponds to five low-levels steps. In the Block-V receiver track ordering constraints can be imposed over a general *calibrate-receiver* activity. However, the Block-IV receiver track contains ordering constraints that refer to lower-level calibration steps. These constraints could be modified to only refer to a more general goal type (consisting of the entire shaded area), but then specialized information would be lost. For instance, in the Block-IV track, it may be desirable to perform MDA configuration (*config-MDA* step) and exciter configuration (*config-exctr* step) in parallel. However, if all ordering constraints are forced to refer to a general calibration goal (and thus moved outside of the shaded area) such parallel execution would violate an ordering constraint.

There are several possible representation solutions for this problem. One solution, which stays entirely within the HTN framework, is to encode separate rules for tracks that require these inter-modular constraints. This would require a main VLBI receiver-calibration rule for each receiver type and constraints common to multiple receiver types would be duplicated in every rule. Unfortunately, this solution results in less rule generality and if these situations are common, the domain definition becomes increasingly complex. For the example mentioned above, we would have to write several sets of rules for the VLBI track, one set that pertained only to Block-IV receivers, and at least one set (possibly several sets) for all other receiver types.

Another possibility is to represent the domain knowledge in a purely operator-based for-

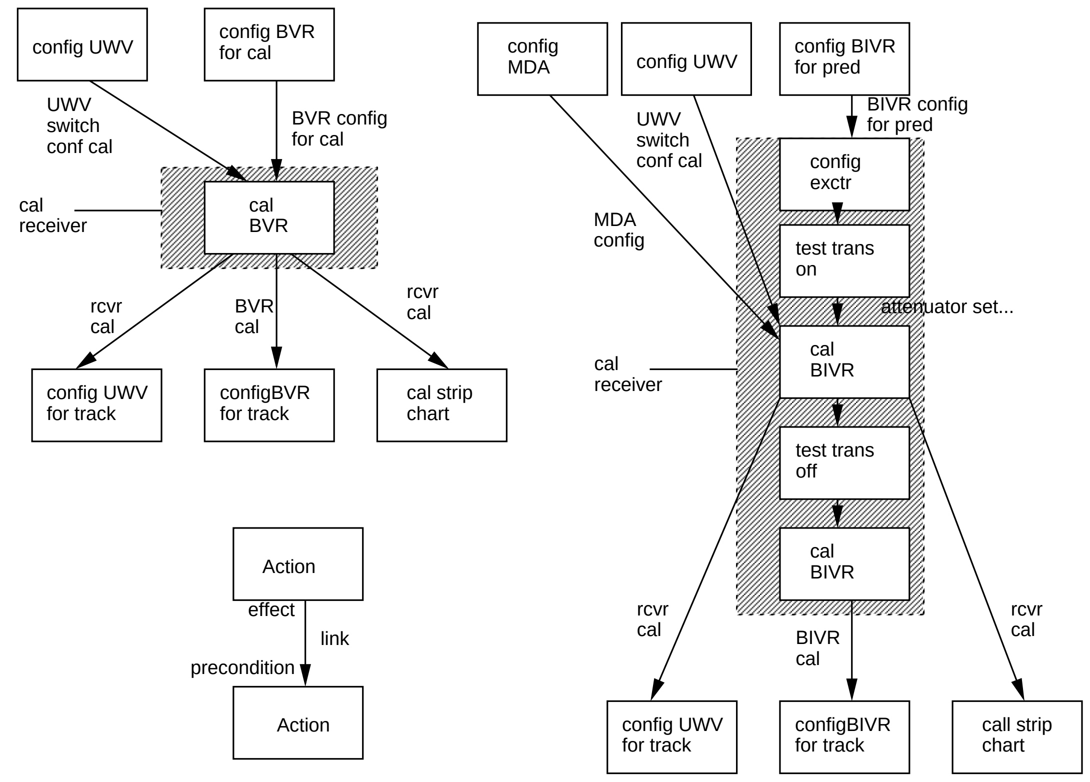

Figure 5: VLBI Receiver Subplans

mat. After all, these problematic ordering constraints are usually due to precondition/effect links between low-level steps. Thus, instead of encoding the domain knowledge using decomposition rules, it would be represented in a STRIPS-like operator format, where each operator contained a set of preconditions and effects. This option often provides a more compact representation of required constraint information; however, it has the disadvantages of (1) losing the representation hierarchy and (2) requiring more search.

A more satisfactory solution is to incorporate operator-based planning techniques with the hierarchical representation. This solution allows us to keep our representation hierarchy, however enables us to easily represent specialized constraints through precondition and effects. Instead of directly adding these constraints to decomposition rules, we can implicitly represent them by adding the appropriate preconditions and effects to low-level track steps. As explained in Section 2.2, it is possible in an integrated framework to specify preconditions and effects for low-level goals. The planner can then impose the appropriate ordering constraints through precondition achievement. This approach permits inter-modular ordering constraints to be separate from decomposition rules thereby allowing rules to retain their modularity. Thus, relevant links, such as the one between the *config-MDA* step and the low-level *calibrate-Block-IV-receiver* step, would be represented through preconditions and effects. The relevant ordering constraints would eventually be added through operator-based precondition achievement. The only drawback to this formulation is that acquiring constraints through goal achievement instead of specifying them directly in decomposition rules increases planning search. However, we feel this is an adequate tradeoff since it allows us to represent our domain information in a more useful and flexible format.

Point 1: Hierarchy and Modularity *HTN approaches have the advantage of easily supporting hierarchical representation of constraints. Constraints can be expressed at an appropriate level of the derivation hierarchy and then inherited. However, specialized constraint information is often difficult to represent in an HTN format. Operator-based approaches have the advantage of generality, since they can cover many planning situations unconsidered by the knowledge engineer. However, an operator-based approach is usually less efficient (i.e. involves more search) and its representation is more difficult to maintain. A hybrid HTN/operator-based approach allows an encoding that supports hierarchy and enhanced generality, without requiring an overly large search space.*

4 Encoding Implicit Constraints

Another advantage to using a hybrid planning system is the ability to encode implicit constraint information. These are constraints that may not be obvious when defining decomposition rules or operators, but are still necessary for correct planning. In an integrated planning system, there are several different approaches to adding constraints. They can be explicitly enforced in decomposition rules, or they can be an emergent property of precondition achievement. Stating ordering constraints in a decomposition rule has the advantage of the constraint being immediately recognized. Constraints added through goal achievement may incur more search since they are not readily apparent; however, using preconditions and effects to represent such constraints can often offer a more compact representation.

Consider the following example. When performing a telemetry pass in the DSN domain, a required step is to position the antenna to point at the specified set of coordinates. This step is represented by the activity *move-antenna-to-point*. However, for many precalibration steps, which prepare the antenna for a transmission, the antenna transmitter is temporarily turned on. For these steps, it is necessary to have the antenna in a stow position where stray transmissions are directed at a harmless location. The antenna is usually not moved to point at the final coordinates until after most precalibration steps have been executed. Unfortunately, when defining the DSN domain, this constraint is often (accidentally) left out of many pre-calibration decomposition rules since it does not directly affect the success of pre-calibration activities.

One way to enforce this constraint through HTN techniques is to explicitly add ordering constraints to all telemetry decomposition rules that specify the activity *move-antenna-to-point* be ordered *after* any activity that turns on the transmitter. This would ensure the constraint is enforced early, thus avoiding extra search. Unfortunately, such a constraint may have to be specified numerous times if there are multiple decomposition rules in which it applies. And if one such constraint is erroneously omitted, it could cause an invalid plan to be produced.

Another possibility is to employ operator-based precondition/effect analysis. We could add a precondition of *antenna-at-stow* to any precalibration activities that could cause antenna transmission. This prevents the *move-antenna-to-point* step from being ordered before any pre-calibration activities that use the transmitter. Unfortunately, this option requires a number of extra preconditions to be added and could possibly induce more search. This could also create a plan with unnecessary activities where the antenna is moved out of stow position too early and then is moved back to stow position before any pre-cal activities that use the transmitter.

The best solution is to use both HTN and operator-based techniques together. In this case we use HTN methods to add an overall track goal of forbidding stray transmissions as a protection during the entire pre-cal process. We then force relevant transmission actions to have a conditional effect which violates this requirement when the precondition*not(antenna-at-stow)* is satisfied.

Point 2: Implicit Constraints *An HTN approach offers great flexibility in specifying arbitrary constraints but may require restating constraints multiple times (when no appropriate hierarchy exists). Operator-based methods can also be used to represent these constraints, however they often leads to a proliferation of operator preconditions and often incur additional search. Hybrid methods offer the greatest flexibility in representing implicit constraints and often provide for a more efficient plan search.*

5 Scripting vs. Declaring

Another notable difference between HTN and operator-based approaches is that an HTN approach allows the encoding of specific (canned) action sequences while an operator-based approach often incurs significant search to construct the same sequence (since the user cannot directly specify it). Conversely, when operators can be combined in different ways but still have interactions, an operator-based representation can be a more concise, natural method of encoding these constraints. In varying domains, or portions of these domains, different aspects of these representation tradeoffs are relevant.

In order to demonstrate this tradeoff we performed an experiment using a knowledge engineer (KE). For this experiment, the KE encoded a simplified portion of the MVP image processing domain[Chien and Mortensen 1996], which represented a subproblem called image navigation.[4] This is perhaps the most complex subproblem in this image processing domain. It involves 8 top-level goals and 40 operators; a typical plan might range from 20-50

[4]The knowledge engineer had some knowledge of the image processing application and had no knowledge of this paper or research topic.

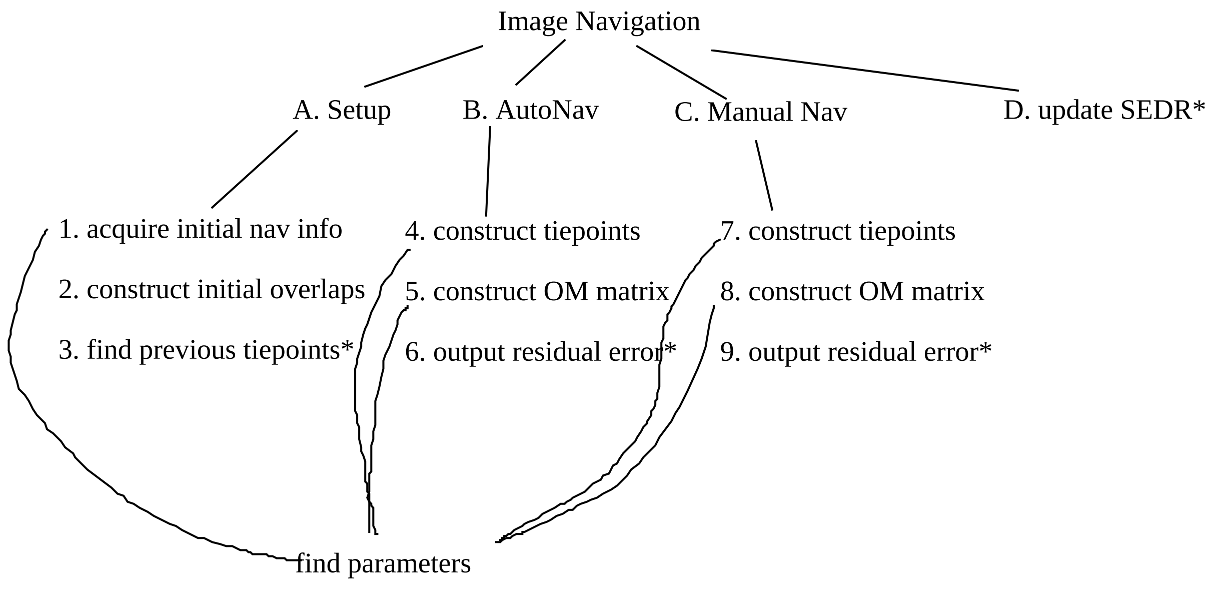

Figure 6: MVP Navigation Process

operators. For this subproblem, the KE developed three planning models, one in which only operator-based techniques were used, one where only HTN techniques were used, and one where both techniques were used.

All possible steps of the image navigation problem are shown in Figure 6. In the most basic case the process would involve setup steps A.1 and A.2, and automatic navigation steps B.4 and B.5. However, in some circumstances all asterisked steps would also be added. For example, if there is an initial tiepoint file, step A.3 might be added. If residual error feedback was desired, step B.6 would be added. And if the user wanted to update the archival SEDR[5], step D would be added. In most situations, the navigation process consists of: A. setup, B. automatic navigation, then C. manual navigation, where steps from manual navigation are performed to fine-tune the results of the automatic navigation.

To even more complicate matters, the exact specification of many steps depends on if other steps are being performed. For example, if residual error output is a requested goal, step B.6 and step C.9 must be executed. This requires that step B.5 and step C.8 have appropriate parameter settings to compute the residual output as part of the OM matrix

[5]Supplementary Experiment Data Record

(the planet-to-camera coordinate transformation matrix) computation.

Furthermore, we have only listed the major component steps of navigating the image. There are also secondary steps that extract information from the image label. These secondary steps help appropriately select program parameters for each of the main steps listed in Figure 6. These extra details account for additional operators and steps in the plan not shown in Figure 6. In all, a complex navigation process would involve a plan of perhaps 30 operators and 50 derivation steps.

We compared the three knowledge bases constructed by the KE for this problem using the following measures: compactness of encoding, modularity (lack of repetition), and search efficiency. In Table 1 we summarize the number of HTN rules, number of operators, and search required for the most complex problems in each of the encodings.

Encoding	# HTN Rules	# Operators	# Search Nodes
Operators	0	8	26
HTN Rules	15	0	5
Hybrid	5	8	18

Table 1: Knowledge Encoding Statistics

Based on our results, the pure operator-based representation was more inefficient from a search perspective and encoding a domain in this type of representation was the most difficult. While only a small subset of the combinations of operators will actually be used in solving problems, this type of framework requires that all operators be sufficiently accurate to rule out all other combinations. Another way to encode this information in the operator-based framework is to add artificial preconditions to help direct the search of the planner. Another less ad hoc method is to specify search control rules to direct the planner. Unfortunately, neither of these methods elegantly and modularly represents the hierarchical scripting aspect of this planning problem. It is difficult to see the few semantically meaningful sequences from looking at the operators; and, it is difficult to debug the operators to ensure generation of only valid sequences. On writing a pure operator-based representation the KE said *"The* *operator KB was the most difficult to encode. One small change typically affected many operators and would require great re-testing. Because I had worked myself into a corner, I had to start from scratch a few times. For the final time, I realized that I needed to fully map the entire structure (including parameters) on paper."*

Representing this problem in a pure HTN framework was also difficult. Many complex combinations of dependencies and interrelations would require numerous decomposition rules. Generally, there is one reduction rule for each basic sequence, and one rule for each combination of add-ons to the basic sequence. For example, when using step B.5, we need to consider the cases where step B.6 was included and the cases where it was not. Also, the OMCOR2 program (step B.5) needs to know whether a previous tiepoint file was used (step A.3), etc. Unfortunately, this creates a proliferation of decompositions rules which are is difficult to understand and maintain. The HTN encoding of this problem resulted in 4 rules to cover the automatic navigation process, 2 to cover the manual navigation process, and also a number of additional rules to address with previous tiepoint files (Step A.3). These rules account for the 15 rules required for the pure HTN representation.

In a combined HTN and operator-based framework it was possible to represent different parts of the plan generation process using operator-based and HTN methods. Basic sequences can be easily represented using HTN rules. More complex additions to each basic sequence can be represented through operator-based constructs such as preconditions and conditional effects. Once the basic sequence has been determined through decomposition, goal-achievement is used to add additional constraints or dependencies. The complex navigation problem discussed above can now be represented as a separate script. For example, the two basic navigation phases, automatic navigation and manual navigation, can now be represented in an HTN framework. However, slight modifications from the default framework (such as whether or not to use an initial tiepoint file) can be linked in using operator-based planning techniques. For example, the required link between step B.5 (OMCOR) and step B.6 (outputting residual error) can be represented by having a conditional effect of the OMCOR operator achieve a precondition of the output residual error step. The presence of this conditional effect represents an additional specification on when to use the OMCOR operator, and thus, also represents the interaction between step B.5 and step B.6. Therefore the basics of the two-pass navigation script are handled by the HTN planner while variations are managed by the operator-based component.

Overall, the use of a combination of HTN and operator-based techniques results in a reduced number of rules (i.e. compactness) and avoidance of redundancy in the KB. Avoiding redundancy is especially important since redundant portions of the KB must all be updated whenever one part is changed. This can lead to errors and increased maintenance costs.

Point 3: Scripting vs Declaring *An HTN framework is more search efficient than an operator-based one in cases where only a few sequences of operators are valid. However, an HTN format may often require numerous rules to represent additional constraints. An operator-based framework is representationally much cleaner, however, it requires a more general set of operators that can correctly manage many possible execution paths. In a hybrid framework, we can interleave the two planning processes (and representations) to produce an efficient planner that supports a compact, maintainable representation.*

6 Other Representational Issues

In this section, we present several other representational issues that often arise when building a planning system for real-world applications.

6.1 Static State Information

One important issue in operator-based and HTN-based planning is the ability to efficiently use static state information to assist in pruning the search space. Often, decomposition conditions or operator preconditions can be considered static if they will remain unchanged throughout the planning process. These conditions can usually be evaluated immediately, which will help to initially prune the search space. For example, consider the DSN decomposition rule and the GALSOS operator (from MVP) shown in Figure 7. In the GALSOS

```
(decomprule PO_required
 lhs
 (initialgoals ((PO_required ?track_id))
  conditions ((CONFIG ?tt PO_FILE ?po_file))
  context   ((c1 (ss_connect ?track_id))
            (c2 (load_ant_predicts ?track_id))))
 rhs
 (newgoals ((g1 (load_PO_files ?track_id ?po_file))
           (g2 (conf_dopp_tuner ?track_id)))))
```

```
operator  GALSOS
   :parameters  ?infile ?ubwc ?calc
    :preconditions
        (project ?infile galileo)
        (raw-data-values ?infile)
   :effects
         (:not (reseaus-intact ?infile))
         (:not (raw-data-values ?infile))
         (:not (missing-lines-filled-in ?infile))
         (radiometrically-corrected ?infile)
         (image-format ?infile halfword)
         (blemishes-removed  ?infile)
  if (UBWC option is selected)
     then (uneven-bit-weight-corrected ?infile)
  if (CALC option is selected)
     then (entropy-values-calculated ?infile)
```

Figure 7: Static Conditions in Operator and Decomposition Rules

operator, the precondition *(project ?infile Galileo)* is considered static, since no domain actions can add or delete the specified project file. Similarly, the DSN decomposition rule contains the static condition *(CONFIG ?tt PO_FILE ?po_file)*. This condition is considered static since the PO file name will always remain constant throughout planning.

Different planners are able to take advantage of this static information in varying degrees. For instance, when the operator-based planner PRODIGY [Carbonell et al. 1992] is considering applying a new operator, it will enumerate all possible variable bindings for any operator arguments. PRODIGY will immediately evaluate a static condition such as *(isa ?x foo)* to help eliminate any inconsistent bindings. While this does help prune the search space, PRODIGY still immediately assigns constants to variables which causes it to unnecessarily commit to bindings for other variables in the operator.

Most partial-order planners, such as UCPOP [Penberthy and Weld 1992], will add all preconditions to an unachieved goal list. Unfortunately, if the preconditions are not attacked in the correct order (e.g. static conditions first), much unnecessary search may occur. Also, even if the static preconditions are immediately attacked, but no bindings can satisfy them, the current (unusable) plan will still be added to the queue and removed (with no new plans generated).

In our integrated planning framework, decomposition rules can use static preconditions to decrease the set of possible variable bindings generated when a rule is applied. Static preconditions are labeled as such and only variable bindings that satisfy them are generated when considering applicable decomposition rules. Thus, codesignation to satisfy static conditions can be committed to early, but unnecessary commitments for other subgoals and variables are avoided. These static conditions are related to filter conditions [Pryor and Collins 1992] since they restrict the applicability of the certain operators. However, precisely because static conditions cannot be changed by operators, they can be easily evaluated and used in determining the applicability of a decomposition rule or operator.

Point 4: Static Conditions *Both operator-based planners and HTN planners can commit on codesignations to evaluate static conditions to increase search efficiency. Hybrid planners can use static information correspondingly.*

6.2 Nominal Planning, Replanning

It may be important in an application to predict (and control) the plans that are generated for nominal or near-nominal conditions. For example, when the problem goals or initial state change slightly, it is often desirable for the output plan to also change only slightly.[6] In operator-based planners, it is often difficult to encode such preferences. The planner would typically only be required to generate a correct plan. In contrast, since HTN planning techniques are closer to scripting, HTN planners offer good control over nominal or near-nominal plan generation. Hybrid HTN/operator planning frameworks can thus also offer control over nominal plan generation.

A key requirement of many real-world planning systems is the ability to replan when plan goals or other conditions change. Replanning generally requires basic knowledge of why certain goals and actions are present in the plan and present in action preconditions and effects. This requires a basic level of operator-based information and is thus mostly supported through operator-based techniques, such as precondition and effect analysis. HTN approaches often encourage the omission of this information from the domain knowledge since

[6]This is a strong user requirement in both the image processing and DSN antenna operations applications.

it is not required for normal planning. Hybrid techniques must still require this precondition and effect information in order to effectively replan. Therefore, if replanning is necessary, much of the ease of encoding in an HTN approach is lost because a significant operator representation is required.

Point 5: Nominal Planning, Replanning *HTN and hybrid approaches also offer greater control over nominal plan generation. However, replanning generally requires operator-based information (preconditions and effects).*

6.3 Goal Regression

In operator-based planning, relevant goal modifiers are listed as arguments to the goal predicate. These modifiers then get propagated from goal to subgoal through operators. Thus, any parameters that are *possibly* relevant to a goal (and any of its subgoals) must be present as goal arguments. [7] This procedure can result in long argument lists (often tens of parameters), thereby increasing the difficulty of knowledge maintenance.

In HTN planning, relevant modifiers are typically propagated top-down from more abstract goals which expand into more specific activities. While this process still requires all possibly relevant parameters to be present, the expansions tend to result in short wide structures (e.g. an HTN rule expands a single goal into many goals). Thus, argument lengths quickly get shorter at lower levels of abstraction.

A hybrid approach requires goal arguments to support both HTN and operator based planning and hence offers no advantage over either.

Point 6: Goal Regression *Operator-based approaches require all relevant goal modifiers be present as goal arguments which results in long argument lists. HTN approaches offer a cleaner representation of complex goal arguments where argument lengths get shorter as goals become more detailed. A hybrid approach is required to support both of these formats.*

[7]Even those parameters relevant for only a few methods of achieving a goal must also be present.

7 Conclusion

This paper has described a number of issues relevant in representing planning knowledge in operator-based and HTN-based planning paradigms, including: (1) modular and hierarchical representation of planning knowledge; (2) declaring versus scripting; (3) encoding control information and implicit knowledge; and (4) additional issues involving utilizing static conditions, nominal planning, and replanning, goal regression. In light of these issues, we have described the main tradeoffs of using an HTN, operator-based, or integrated specification to represent domain knowledge. In particular, we discuss how these different methodologies impacts the naturalness of the representation using the evaluation criteria of generality, hierarchy, flexibility, and efficiency.

HTN and hybrid approaches are strong at modular and hierarchical representation, however operator-based approaches usually provide a more compact representation of constraints. Hybrid representations are best at managing the tradeoff between generality and efficiency. Hybrid approaches are also most flexible at encoding implicit constraints. All three approaches can incorporate knowledge of static conditions to improve efficiency. HTN/hybrid approaches offer the most control over nominal plan generation, but operator-based techniques offer the most support for replanning. HTN approaches most cleanly represent goal argument regressions. Based on these criteria we conclude that neither the operator-based approach nor the HTN approach dominates the other. Rather, in some cases the operator-based representation is more appropriate and in other cases the HTN representation is more appropriate. Thus, it seems most prudent to advocate usage of hybrid HTN/operator techniques.

Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service

by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[Carbonell et al. 1992] Carbonell, J.G.; Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Pérez, M. A.; Reilly, S.; Veloso, M.; and Wang, X. 1992. PRODIGY *4.0: The Manual and Tutorial.* Technical report, School of Computer Science, Carnegie Mellon University.

[Chapman 1987] D. Chapman, “Planning for Conjunctive Goals”, 1987, *Artificial Intelligence 32*, 3, pp.333-377.

[Chien 1994] S.Chien,”Automated Synthesis of Image Processing Procedures for a Large-scale Image Database,” Proceedings of the First IEEE International Conference on Image Processing, Austin, TX, November 1994, Vol. 3, pp. 796-800.

[Chien and Mortensen 1996] S. A. Chien and H. B. Mortensen, ”Automating Image Processing for Scientific Data Analysis of a Large Image Database,” IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (8): pp. 854-859, August 1996.

[Chien et al. 1995] S. A. Chien, R. W. Hill Jr., X. Wang, T. Estlin, K. V. Fayyad, and H. B. Mortensen, “Why Real-world Planning is Difficult: A Tale of Two Applications,” Proceedings of the Third European Workshop on Planning, Assisi, Italy, September 1995.

[Chien et al. 1997] Chien, S., Govindjee, A., Estlin, T., Wang, X., Hill Jr., R., ”Using Artificial Intelligence Planning Techniques to Automate Generation of Tracking Plans for a Network of Communications Antennas,” Proceedings of the 1997 Conference on Innovative Application of Artificial Intelligence, Providence, RI, July 1997, pp.963-970.

[Erol et al. 1994] K. Erol, J. Hendler, and D. Nau, "UMCP: A Sound and Complete Procedure for Hierarchical Task Network Planning," Proceedings of the Second International Conference on AI Planning Systems, Chicago, IL, June 1994, pp. 249-254.

[Firby 1996] J. Firby, "Modularity Issues in Reactive Planning," Proceedings of the Third International Conference on AI Planning Systems, Edinburgh, UK, May 1996, pp. 78-85.

[IEEE Expert 1996] AI Planning Systems in the Real World, IEEE Expert, December 1996, pp. 4-12.

[Kambhampati 1995] Kambhampati, S., A Comparative Analysis of partial order planning and task reduction planning, SIGART Bulletin, Special Issue on Evaluating Plans, Planners, and Planning, Vol 6, No. 1, January 1995, pp.16-25.

[Minton et al. 1991] Minton S., J. Bresina, and M. Drummond, "Commitment Strategies in Planning: A Comparative Analysis," Proceedingsof the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, pp.259-267.

[Penberthy and Weld 1992] J. S. Penberthy and D. S. Weld, "UCPOP: A Sound Complete, Partial Order Planner for ADL," Proceedings of the Third International Conference on Knowledge Representation and Reasoning, Cambridge, MA October 1991, pp. 103-114.

[Pryor and Collins 1992] G. Collins and L. Pryor, "Achieving the functionality of filter conditions in a partial order planner," Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, CA, July 1992, pp. 375-380.

[Tate et al. 1994] Tate, A., B. Drabble, and R. Kirby, "O-Plan2: An Open Architecture for Command Planning and Control," in *Intelligent Scheduling* (Eds. M. Fox and M. Zweben), Morgan Kaufmann, 1994, pp. 213-239.

[Wilkins 1988] D. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, 1988.